Motion planning with constraints for a safe human-robot interaction

Pol Ramon-Canyameres
Inst. of Industrial and Control Eng.
Universitat Politècnica de Catalunya
Barcelona, Spain
pol.ramon@upc.edu

Isiah Zaplana
Inst. of Industrial and Control Eng.
Universitat Politècnica de Catalunya
Barcelona, Spain
isiah.zaplana@upc.edu

Jan Rosell
Inst. of Industrial and Control Eng.
Universitat Politècnica de Catalunya
Barcelona, Spain
jan.rosell@upc.edu

***Abstract*—Motion planning for robotic manipulators in human-shared environments demands strategies that account for human safety. The consideration of artificial constraints imposed to the planner may be a good alternative to limit the space and the types of motions the robot is allowed to do. In this line, this paper presents a constraint-based motion planning framework designed to facilitate the definition, visualization, and combination of task-specific constraints. The capabilities of the framework are demonstrated through experiments with a dual-arm robot operating in a shared workspace with a human, by defining constraints tailored to human-robot interaction tasks like hand-over actions. The proposal is built on sampling-based techniques and efficiently implemented using the Open Motion Planning Library (OMPL). The integration of the framework into a knowledge-based planning system using ontologies is under development, in order to automatically determine the constraints to be used in each situation to easily improve safety and comfort in human-robot collaboration.**

***Index Terms*—robotic manipulation, human-robot interaction, motion planning, task-specific constraints**

I. Introduction

The planning of motions for robotic manipulators has been an active research field in recent decades, with very good results mainly attributed to sampling-based approaches, such as Probabilistic Roadmaps (PRMs) or Rapidly-exploring Random Trees (RRTs). Many challenges have been faced, besides the finding of collision-free paths, such as the need to consider task constraints, to work in dynamic and uncertain environments, or to consider the presence of humans sharing the environment or actively collaborating with the robot. Regarding human-robot interaction, different strategies have been proposed to ensure safety while maintaining task effectiveness in shared workspaces. For instance, some of them include incorporating human-related constraints into cost-space planning algorithms based on T-RRTs [1], using a variant of RRT-Connect that considers a composite cost function that includes several criteria such as human separation distance or visibility [2], or implementing real-time obstacle tracking using potential field methods for collision avoidance preserving task constraints [3]. Experimental evaluations have shown that such safety-oriented motion planning can significantly reduce user anxiety and surprise during human-robot interactions [4], which has also led to proposals directed at making robots move in a human-like manner [5].

Overall, these approaches demonstrate the importance of considering human factors and geometric constraints in developing safe and effective motion planning algorithms for human-robot collaboration. In this line, this paper proposes a motion planning framework to facilitate the definition of task constraints (particularized to human-robot cooperative tasks in shared workspaces), the efficient computation of solutions satisfying these constraints (by using sampling-based methods implemented by the Open Motion Planning Library [6]), and the application to a particular dual-arm robot. The contributions of this framework for constraint-motion planning can be summarized as follows:

- Easy way to define and combine task constraints, and to visualize them.
- Constraints tailored to cooperative tasks between humans and robots.
- Efficient implementation based on the OMPL.

The remainder of this paper is organized as follows. After this introduction, Sections II and III present some preliminary concepts, tools, and the experimental setup used for the implementation and validation of the proposal. Section IV presents the proposed approach, which is validated in section V. Finally, Section VI presents conclusions and future work.

II. Preliminaries

Sampling-based motion planning methods have emerged as one of the most effective approaches for computing collision-free paths for robotic manipulators, particularly due to their ability to manage high-dimensional configuration spaces. Research in this area has primarily focused on two foundational methods: the Probabilistic Roadmap Method (PRM [7]) and the Rapidly-exploring Random Tree (RRT [8]). Both methods rely on sampling configurations within the robot's configuration space and connecting them to form either roadmaps (as in PRMs) or trees (as in RRTs), thereby capturing the connectivity of the free-of-obstacles configuration space. Numerous variants have been developed to enhance efficiency by biasing the sampling toward more promising regions, optimizing specific performance metrics, or incorporating constraints related to the task or the robot itself.

When motion planning problems involve geometric task constraints, such as requiring the robot end-effector to remain on a surface, follow a path, or maintain a given orientation,

the set of valid solutions is confined to a lower-dimensional manifold embedded in the configuration space. In this context, standard sampling-based methods are insufficient, as randomly drawn configurations are unlikely to lie on this task-constrained manifold. To address this, some constraint-aware extensions of sampling-based planners have been proposed that incorporate projection techniques [9], i.e. candidate samples are first generated in the ambient space and then projected onto the constraint manifold using numerical methods (e.g., iterative solvers, Jacobian-based projections). This ensures that only configurations satisfying the constraints are included and connected, enabling the planner to construct valid paths that lie entirely within the feasible task-constrained subset of the configuration space.

In collaborative human-robot tasks, motion planning must ensure that robot trajectories are not only collision-free but also guarantee safety and increase trust. For this, planned motions must comply with several types of constraints: physical constraints (such as joint and velocity limits, or coupled joint motions that may be structural or software-imposed to produce human-like movements); spatial constraints (such as maintaining a required clearance, enforcing specific end-effector orientations, or restricting the end-effector to a task-dependent volume or predicted human occupancy space); and human-centered constraints (such as ensuring that robot motions are visible, predictable, and interpretable by human partners).

The framework proposed in this paper employs a constraint-based motion planning approach built on RRTs. It incorporates constraints that allow maintaining the robot end-effector at a fixed orientation (with adjustable tolerance to constraint satisfaction, if necessary), as well as several types of volume constraints. Both individual constraints and their combinations can be used to define task-specific requirements that ensure safety in human-robot collaborative scenarios.

III. Tools

A. The Open Motion Planning Library

The Open Motion Planning Library (OMPL, [6]) is a C++ open-source software library specifically designed for robot motion planning using sampling-based algorithms such as RRT, PRM, and many variants. The library is focused in efficiently implementing all planning algorithms, relying instead on external systems to define the state-space and collision-checking mechanisms. The OMPL includes a constrained-motion planning extension that addresses scenarios in which a robot's motion is restricted to lie on a lower-dimensional manifold defined by constraints (e.g., end-effector poses on a surface or maintaining a fixed distance). This extension supports both equality and inequality constraints and provides multiple strategies for sampling, interpolation, and local planning under constraints, as well as planning algorithms such as projection-based planners that project sampled states back onto the manifold, making it highly suitable for any task requiring compliance with complex kinematic or task-specific constraints.

B. The Kautham Project

The Kautham Project [10] is an open-source framework designed for motion planning in robotics, integrating a wide range of planning algorithms and tools to facilitate research and development in this field. It is built on top of the OMPL, providing the modelling of robots and the environment, the visualization and the collision-checking procedures. It also uses the capabilities of ROS (Robot Operating System) for communication purposes, implemented in a ROS package called $kautham\_ros$. Kautham allows users to define complex planning problems using XML files: a) *problem files*, that describe robots and obstacles within customizable environments, and b) *control files*, that describe which are the controlled joints and how they are actuated, i.e., one control can actuate one or several (coupled) joints. It supports both single and multi-robot systems, offering a graphical user interface to design and simulate scenarios interactively.

C. The Mobile Anthropomorphic Dual-Arm Robot

The Mobile Anthropomorphic Dual-Arm Robot MADAR [11] shown in Fig. 1 serves as an experimental platform for evaluating the constraint motion planning framework proposed in this study. By emulating human morphology, MADAR is designed for manipulation tasks.

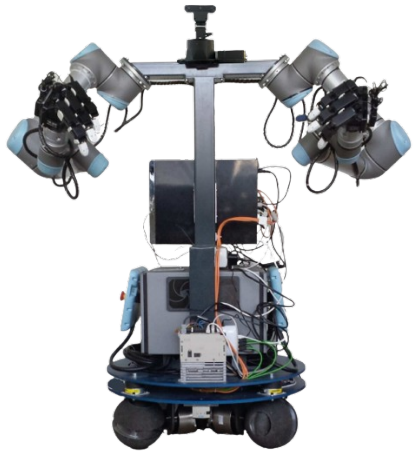

Fig. 1: The experimental robot, MADAR [11].

MADAR consists of an omnidirectional mobile base with three wheel-motor units spaced 120° apart, incorporating LiDAR sensors and vibration dampening. On top of the base, two UR5 robotic arms (six DOFs, 5 kg payload) are mounted on a T-shaped structure. Each arm is equipped with an Allegro V4 hand featuring four fingers with torque-controlled joints and fingertip tactile sensors. A pan-tilt structure supports a camera, providing a movable field-of-view for visual tasks. The configuration of the arms ensures a good workspace overlap and high manipulability in front of the robot, where bimanual tasks are primarily executed.

All the components are managed by an on-board CPU using C++ and ROS 2 [12], which facilitates real-time integration and communication. A dedicated ROS node for the

base converts velocity commands into motor control signals for the wheels. The arms are managed by a separate node that supports both trajectory execution and end-effector pose commands. The hand node handles communication and kinematic computations, supporting multiple grasping modes and control strategies. The head is controlled by a node that enables target tracking within the camera's field of view. A high-level ROS node coordinates the synchronized execution of multisubsystem movements, whereas the head subsystem operates independently.

D. The Motion Manager

A ROS node has been implemented as a manager to orchestrate and streamline the communication between *kautham_ros* and the MADAR ROS nodes. This manager node also includes a user interface for defining the problem and for monitoring and controlling the system. Fig. 2 shows the communication between nodes, where the ROS service calls are detailed.

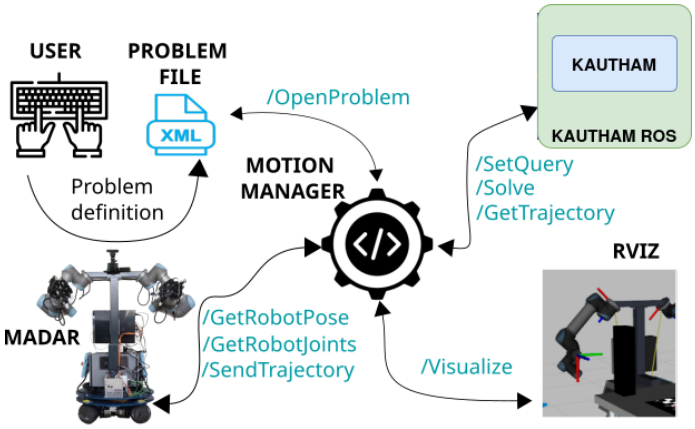

Fig. 2: The Motion Manager orchestrates the communications between the motion planner and the real system. In blue, the ROS services used between nodes.

IV. Proposed Solution

In this section, a detailed description of the key components that constitute the developed constraint-based motion planning framework is provided. Each component plays a critical role in the definition, management, and execution of constraints within the planning process. In particular, the following subsections address: (1) how to define constraints in Kautham, including the configuration of their parameters and properties; (2) the types of constraints available; and (3) the application of constraints during the motion planning process.

A. Kautham constraints

The `Kautham Constraint` class inherits from the `ompl::base::Constraint` class and provides additional functions that allow users to define new, customizable constraints with minimal programming effort. In particular, some of the properties users can customize are:

- *The constrained joints*. These are the joints that will be used to satisfy the defined constraints. These joints must be consecutive joints of a kinematic chain and the joints that precede them must not belong to the set of controlled joints, i.e., the transform between the robot base frame and the link prior to the first constrained joint is considered fixed.
- *The target link*. It specifies the robot link to which the constraint is applied. The `target_link` at least must correspond to the last link actuated by a constrained joint or to a subsequent link in the kinematic chain provided that none of the joints located between the last constrained joint and this link is a controlled joint, i.e., the transform between the last link actuated by a constrained joint and the `target_link` must be fixed.
- *The constraint logic*. This logic is implemented by overriding the `function()` method in the `ompl::base::Constraint` class. This function encodes the constraint and defines the conditions under which it is considered satisfied. In particular, if the constraint is represented by a function $f : \mathcal{C} \rightarrow \mathbb{R}^n$, where $\mathcal{C}$ denotes the configuration space of the robot and $\mathbb{R}^n$ is an n-dimensional vector space, then the constraint is considered fulfilled whenever $f(\boldsymbol{q}) = 0$.
- *The dynamic tolerance*. To introduce flexibility into the problem, the `Kautham Constraint` class provides an enhanced mechanism for specifying the constraint tolerance by allowing the use of dynamic tolerance values. Specifically, the tolerance can be defined as a function of the translational distance between the current and goal positions of the `target_link` in the three-dimensional Euclidean space. This dynamic tolerance is characterized by a nominal value, which applies at the goal, and a gradient that increases the allowable tolerance as the distance from the goal position increases. This formulation results in a spherical region centered at the goal position of the `target_link`, within which the constraint becomes progressively stricter as the end-effector approaches the goal, ultimately restricting to the nominal tolerance value at the target position.

B. Types of constraints

In Kautham, a set of predefined constraints has been implemented, each accessible through a unique `type` tag identifier. When Kautham constructs the internal representation of the problem, it automatically loads the requested constraint from the `Constraint Factory`, where all implemented constraints must be defined and associated with their respective `type` tag identifier. They are the following:

- **Orientation constraints:** Defined to enforce that the end-effector maintains a desired orientation, defined relative to the robot's base link, throughout the motion planning process. This is achieved by computing the rotational error between the current and target orientations. The allowable tolerance for this error can be dynamically adjusted based on the distance to the goal, if specified. In addition, this type of constraint supports selective enforcement along specific axes defined in the target link

frame; that is, instead of applying the constraint to all three orientation components, it can be limited to one or two. This approach enables more flexible control of the end-effector's orientation, allowing constraints to be tailored to the requirements of specific orientation-related tasks.

- **Geometric constraints:** Defined to enforce that the robot's end-effector position remains within (or outside) a specified geometric region, expressed (as shown in Fig. 3) relative to the base link of an obstacle in the environment, throughout the motion planning process. For each geometric constraint, the algorithm evaluates the extent to which the current pose violates the geometric boundary.

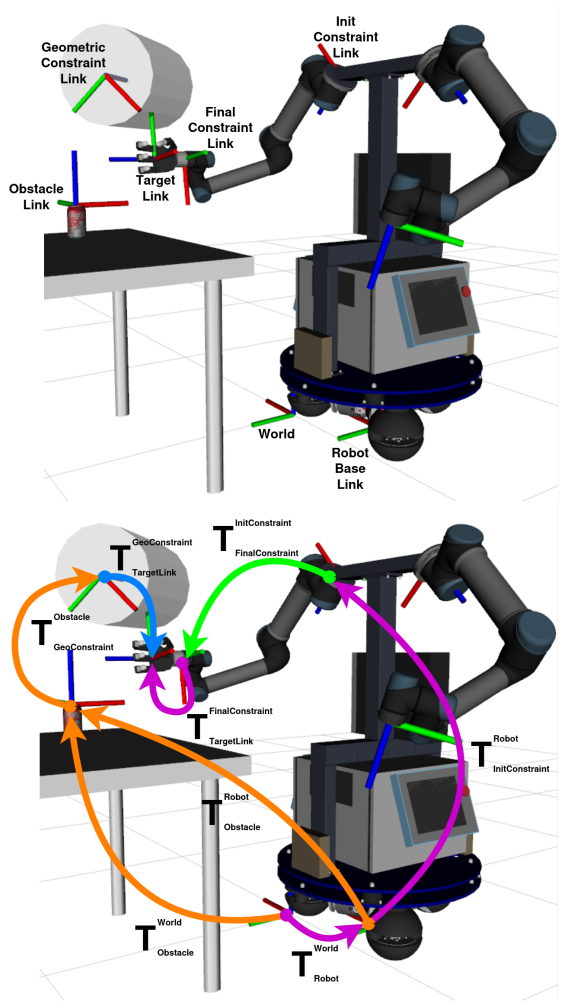

Fig. 3: Representation of the frames (top figure) and transformations (bottom figure) associated with geometric constraints.

Several types of geometric constraints have been implemented. They can be described as follows, assuming the "within" formulation:

a) Box constraint: The end-effector must remain inside a box of dimensions (l, w, h), which is encoded by the function $f = (f_{\text{length}}, f_{\text{width}}, f_{\text{height}})$, with:

$$f_{\text{length}}(\boldsymbol{q}) = \begin{cases} 0, & \text{if } |x| \leq \frac{l}{2} \\ \text{sgn}(x)\left(|x| - \frac{l}{2}\right), & \text{otherwise} \end{cases}$$
$$f_{\text{width}}(\boldsymbol{q}) = \begin{cases} 0, & \text{if } |y| \leq \frac{w}{2} \\ \text{sgn}(y)\left(|y| - \frac{w}{2}\right), & \text{otherwise} \end{cases} \quad (1)$$
$$f_{\text{height}}(\boldsymbol{q}) = \begin{cases} 0, & \text{if } |z| \leq \frac{h}{2} \\ \text{sgn}(z)\left(|z| - \frac{h}{2}\right), & \text{otherwise} \end{cases}$$

where (x, y, z) are the coordinates of the end-effector's position, when the robot is at configuration $\boldsymbol{q}$, expressed with respect to the reference frame located at the box center.

b) Sphere constraint: The end-effector position must remain inside a sphere of radius r, which is encoded by the function f:

$$f(\boldsymbol{q}) = \begin{cases} 0 & \text{if } \|p(\boldsymbol{q})\| \leq r \\ \|p(\boldsymbol{q})\| - r & \text{if } \|p(\boldsymbol{q})\| > r \end{cases} \quad (2)$$

where $\|p(\boldsymbol{q})\|$ is the Euclidean distance between the end-effector's position, when the robot is at configuration $\boldsymbol{q}$, and the origin of the sphere.

c) Cylinder constraint: The end-effector must remain within a cylinder of radius r and height h, which is encoded by the function $f = (f_{\text{radius}}, f_{\text{height}})$, with:

$$f_{\text{radius}}(\boldsymbol{q}) = \begin{cases} 0 & \text{if } \sqrt{x^2 + y^2} \leq r \\ \sqrt{x^2 + y^2} - r & \text{otherwise} \end{cases}$$
$$f_{\text{height}}(\boldsymbol{q}) = \begin{cases} 0 & \text{if } 0 \leq z \leq h \\ z & \text{if } z < 0 \\ z - h & \text{if } z > h \end{cases} \quad (3)$$

where (x, y, z) are the coordinates of the end-effector's position when the robot is at configuration $\boldsymbol{q}$, expressed with respect to the cylinder's reference frame, which is located at one of its bases with the z-axis aligned with the height and pointing towards the cylinder's interior.

d) Cone Constraint: The end-effector must remain within a cone of height h and base radius r, which is encoded by the function $f = (f_{\text{radius}}, f_{\text{height}})$, with:

$$f_{\text{radius}}(\boldsymbol{q}) = \begin{cases} 0, & \text{if } R \leq R_{\max},\ 0 \leq z \leq h \\ R - R_{\max}, & \text{if } R > R_{\max},\ 0 \leq z \leq h \\ \infty, & \text{otherwise} \end{cases}$$
$$f_{\text{height}}(\boldsymbol{q}) = \begin{cases} 0 & \text{if } 0 \leq z \leq h \\ z & \text{if } z < 0 \\ z - h & \text{if } z > h \end{cases} \quad (4)$$

where $R_{\max} = \frac{z}{h} r$, $R = \sqrt{x^2 + y^2}$ and (x, y, z) are the coordinates of the end-effector's position, when the robot is at configuration $\boldsymbol{q}$, expressed with respect to the reference frame, which is located at the cone apex with

the z-axis aligned with the height and pointing towards the cone's interior.

- **Constraints combination:** The OMPL has the class `ompl::base::ConstraintIntersection` that allows multiple constraints to be combined into a single composite constraint, which is satisfied only when all individual constraints are simultaneously satisfied. Kautham automatically manages the combination of constraints by grouping them according to their constrained joint names. This grouping ensures that only constraints affecting the same set of joints are combined. When multiple constraints are defined in a single robot, without the intention to be combined, it is not possible to involve the same joint names across different constraints, unless the entire set of joints is identical for those combined constraints. This restriction ensures that each joint is subject to only one constraint within any given combination or set, thereby avoiding conflicts during the motion planning.

C. Applying constraints to the motion planning

When defining a motion planning problem, some constraints can be included and assigned to a robot by defining them in the Kautham problem file within the robot's XML tag, as illustrated in Figs. 4 and 5.

```
<Constraint id="ori_arm_right" type="arm_orientation">
    <Enabled status="true"/>
    <TargetLink name="right_wrist_3_link"/>
    <FreeMovementAxes x="false" y="false" z="false"/>
    <TargetOrientation qx="-0.500" qy="0.500" qz="0.500" qw
     ="0.500"/>
    <Tolerance value="0.1" variable="false" gradient="0.0"/
     >
    <Joint name="right_shoulder_pan_joint" />
    <Joint name="right_shoulder_lift_joint" />
    <Joint name="right_elbow_joint" />
    <Joint name="right_wrist_1_joint" />
    <Joint name="right_wrist_2_joint" />
    <Joint name="right_wrist_3_joint" />
</Constraint>
```

Fig. 4: Example of an orientation constraint defined as part of a Kautham problem file.

```
<Constraint id="cone_arm_right" type="cone">
    <Enabled status="true"/>
    <TargetLink name="right_wrist_3_link"/>
    <GeometricParams radius="0.2" height="1.0"/>
    <ReferenceFrame entity="coke_can" link="base"/>
    <Origin xyz="0.264 -0.800 0.195" rpy="1.57 1.0 0.0"/>
    <Joint name="right_shoulder_pan_joint" />
    <Joint name="right_shoulder_lift_joint" />
    <Joint name="right_elbow_joint" />
    <Joint name="right_wrist_1_joint" />
    <Joint name="right_wrist_2_joint" />
    <Joint name="right_wrist_3_joint" />
</Constraint>
```

Fig. 5: Example of a cone constraint defined as part of a Kautham problem file.

Kautham allows control over different subsets of joints, but always operates within a single configuration space, $\mathcal{C}$. This design allows users to solve multiple motion planning problems using the same problem file, without rebuilding the internal configuration space each time, thereby saving computational resources. For this reason, all potential constraints must be defined in the Kautham problem file in advance; however, they can be selectively enabled or disabled as needed, and their parameters can be updated. When multiple constraints of the same type are defined for a single robot, matching one of the implemented constraint types, each must be assigned a unique identifier. The `<Joint>` nodes define which joints are going to be used to satisfy the constraint, and their order must follow the robot's kinematic chain.

As shown in Fig. 4, an orientation constraint is applied to the `right_wrist_3_link` of the MADAR robot, with the target orientation specified using quaternions and applied across all three rotational axes (i.e., a full orientation constraint). Similarly, as illustrated in Fig. 5, the cone geometric constraint is also applied to the `right_wrist_3_link` of the MADAR robot, with its vertex expressed relative to the base link of the obstacle `Coke can` via a specified transformation. Since both examples constrain the same set of joints, as explained before, Kautham will combine both.

V. Experimental validation

Two experiments were conducted to validate the use of constraints in motion planning for safe human-robot interaction. All experimental videos are available online through the provided link [1].

A. Handling task

In this experiment, a human shares a workspace with the robot. The human holds a filled Coke can, and the robot is tasked with taking the can from the human's hand and placing it on the table. The constraints imposed to the MADAR and included in the Kautham problem file are defined as follows:

- An orientation constraint to ensure that the filled Coke can remains upright during handling.
- A box-shaped volume to define the shared workspace where the interaction occurs, as well as to ensure that the human remains outside this area for safety.
- A cone constraint, located within the shared workspace, to guide the robot to approach the human hand safely.

Once the internal joint space configuration is established, the Motion Manager updates the Kautham environment to reflect the real environment using the perception system mounted at the MADAR head. This step is crucial to enable adaptive planning during the interaction.

The initial query is then set according to the current joint configuration of the robot, followed by setting the goal query. To define the goal query, the updated obstacle pose is retrieved, and the optimal grasp pose is computed based on the task requirements. As a result, a target link pose is determined,

[1] https://sir.upc.edu/projects/smarthandle/etfa2025

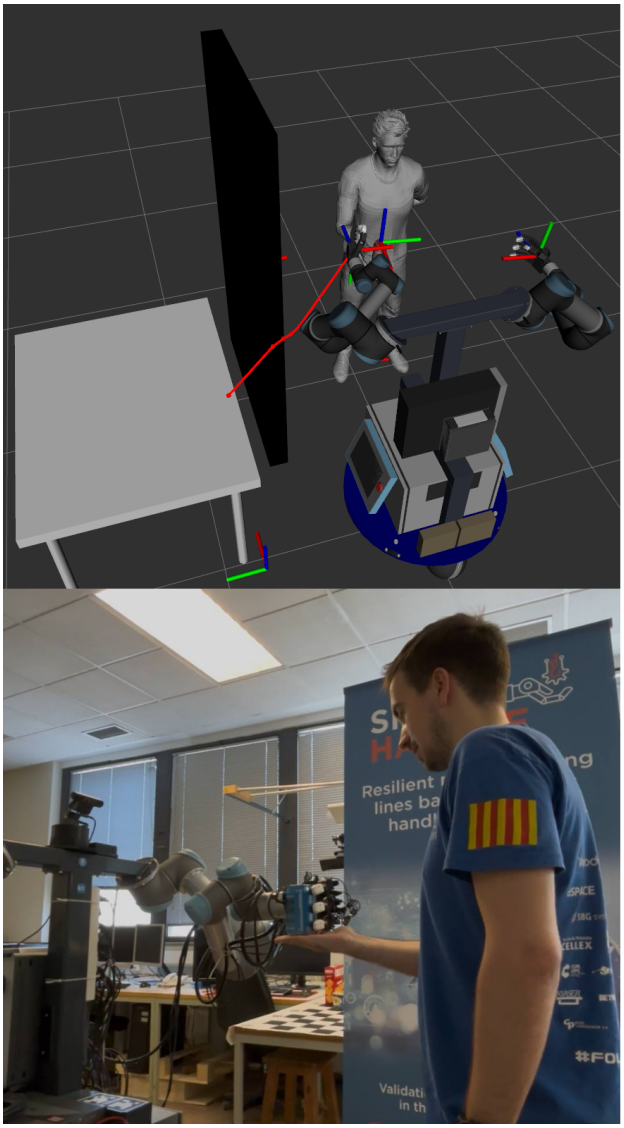

Fig. 6: Motion planning solution trajectory for the handling task (top figure) and real execution of the task (bottom figure).

and the inverse kinematics (IK) of this pose is calculated to set that joint configuration as the goal query.

Subsequently, the orientation constraint is set according to the updated orientation of the Coke can, the origin of the cone constraint is updated according to the updated position of the Coke can, and the cone constraint orientation is set to align with the current end-effector pose.

Once the problem is formulated and the environment configuration remains unchanged, Kautham solves the motion planning problem and returns a valid trajectory, which is subsequently executed by the robot.

Upon reaching the goal, the robot grasps the object. The process is then repeated to place the object on the table while maintaining the required orientation and ensuring safe human–robot interaction by planning motions that avoid obstacles in the environment.

Fig. 6 shows a screenshot illustrating the motion planning result produced by the Kautham tool (from the current configuration to the goal grasping configuration) and a video frame from the real-world experiment, showing the robot grasping the object while executing the computed trajectory on the real robot.

B. Supervised Peg-in-Hole task

In this experiment, a human supervises the robot as it performs a peg-in-hole task using both robotic manipulators. The robot must complete the task without entering the safe volume defined around the human supervisor. The constraints imposed to the MADAR and included in the Kautham problem file are defined as follows:

- An orientation constraint, with dynamic tolerance, to ensure proper alignment of both items during the insertion.
- A cylinder-shaped volume to define the safe area surrounding the human.

The experiment begins with the perception system updating the environment model, including the positions of the person, the peg, and the hole objects, the latter two being grasped by each robot manipulator. The initial configuration of the robot is set based on the current joint configuration of each manipulator. The goal configuration is determined by computing the optimal poses for both end-effectors to achieve precise peg alignment and insertion, while maintaining the required orientation constraints.

During planning, the Kautham tool ensures that the planned trajectories for both arms do not violate the cylindrical safe volume around the person, thus guaranteeing human safety. The motion planner computes a collision-free and constraint-compliant trajectory for the bi-manual manipulation task.

After a valid trajectory is found, the robot executes the peg-in-hole task under human supervision. The entire process is monitored to ensure the robot maintains safe distances and proper alignment throughout the operation.

Fig. 7 shows a screenshot illustrating the motion planning result produced by the Kautham tool, where the cylindrical constraint that protects the supervisor of the task is shown.

VI. Conclusions and Future Works

This paper presents a constraint-based motion planning framework designed for human-robot collaboration in shared workspaces. The framework allows for the intuitive specification of constraints and the efficient computation of trajectories that satisfy them, using constraint-based motion planning algorithms from the Open Motion Planning Library (OMPL). It supports orientation and volume constraints, as well as their combinations, which are particularly suited for tasks such as hand-overs operations in which human-operators are present or intervene. The use of RViz facilitates the visualization of both constraints and the resulting paths. A motion manager has also been developed to interface with the motion planner and compute the trajectory, based on the planned path, for execution on the physical robot. This allows for the demonstration of

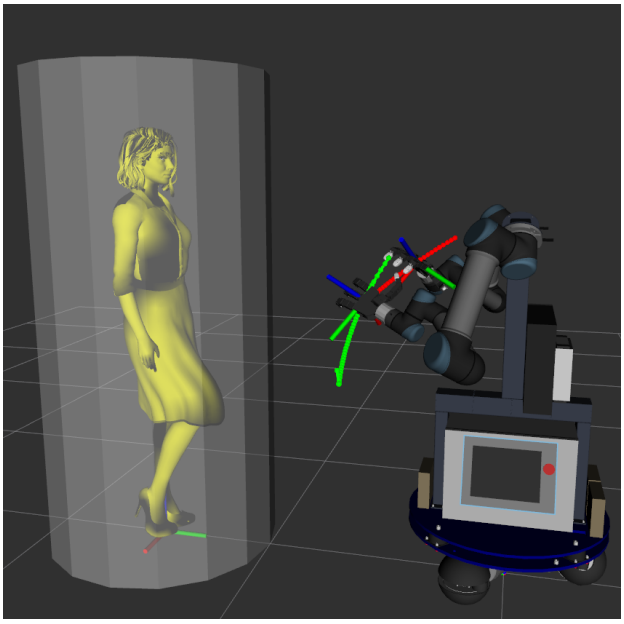

Fig. 7: Motion planning solution trajectory for the supervised peg-in-hole task.

the framework's practical capabilities using the MADAR dual-arm mobile robot, showcasing its applicability to real-world collaborative scenarios.

As a forward-looking step, the framework is planned to be enhanced by incorporating semantic reasoning capabilities through the use of ontologies [13]. Ontologies enable the structured representation of task, environment, and human-related knowledge, providing a formal basis to automatically infer appropriate motion constraints depending on the context of the human-robot task. Furthermore, the integration of Large Language Models (LLMs) is currently being explored to support the semi-automatic or automatic population of these ontologies using domain knowledge, user descriptions, or procedural task definitions [14]. This synergy between symbolic reasoning and generative AI has the potential to greatly simplify constraint specification to be used by this framework and to increase the autonomy and adaptability of robotic planning systems in complex, dynamic environments.

ACKNOWLEDGMENT

This work was funded by the European Union with the project Resilient manufacturing lines based on smart handling systems (SMARTHANDLE) under Grant Agreement 101091792.

REFERENCES

[1] J. Mainprice, E. Akin Sisbot, L. Jaillet, J. Cortes, R. Alami, and T. Simeon, "Planning human-aware motions using a sampling-based costmap planner," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 5 2011. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2011.5980048

[2] V. Rajendran, P. Carreno-Medrano, W. Fisher, and D. Kulić, "Human-aware rrt-connect: Motion planning for safe human-robot collaboration," in *2021 30th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2021, pp. 15–22.

[3] J.-H. Chen and K.-T. Song, "Collision-Free Motion Planning for Human-Robot Collaborative Safety Under Cartesian Constraint," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2018, pp. 4348–4354. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2018.8460185

[4] D. Kulic and E. Croft, *Safe Motion Planning for Human-Robot Interaction: Design and Experiments*. I-Tech Education and Publishing, dec 1 2006. [Online]. Available: http://dx.doi.org/10.5772/4689

[5] N. García, J. Rosell, and R. Suárez, "Motion planning by demonstration with human-likeness evaluation for dual-arm robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2298–2307, 2019.

[6] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[7] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[8] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[9] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *Intl. J. of Robotics Research*, vol. 38, no. 10–11, pp. 1151–1178, Sep. 2019.

[10] J. Rosell, A. Pérez, A. Aliakbar, Muhayyuddin, L. Palomo, and N. García, "The Kautham project: A teaching and research tool for robot motion planning," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8.

[11] R. Suárez, L. Palomo-Avellaneda, J. Martinez, D. Clos, and N. García, "Development of a dexterous dual-arm omnidirectional mobile manipulator," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 126–131, 2018, 12th IFAC Symposium on Robot Control SYROCO 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589631833235X

[12] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[13] O. Ruiz-Celada, A. Dalmases, I. Zaplana, and J. Rosell, "Smart perception for situation awareness in robotic manipulation tasks," *IEEE Access*, vol. 12, pp. 53 974–53 985, 2024.

[14] V. Molina, O. Ruiz-Celada, R. Suarez, J. Rosell, and I. Zaplana, "Robot situation and task awareness using large language models and ontologies," in *Accepted to the 1st International Workshop on Safe and Sustainable AI-Aided Manufacturing (S2AIM)*, 2025.